HOME ALARM SYSTEM
PROTECTING YOUR HOME

Work By:
Alessandro Nardin
Bilal Soussane
Kuba Di Quattro
Niccolò Lechthaler

PROBLEM

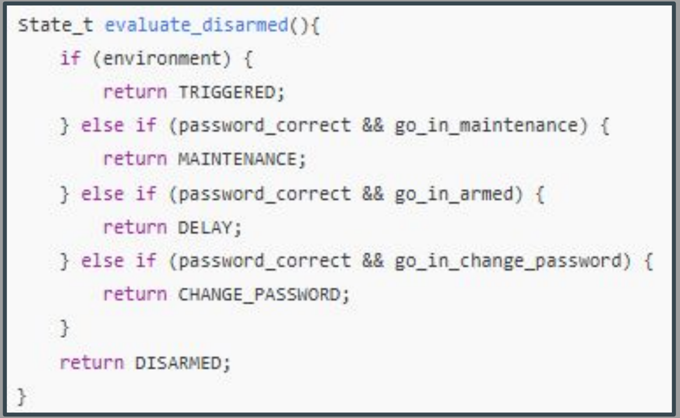
- BURGLARY
- FIRE
- SYSTEM TAMPERING → REMOVING COVER OF ALARM BOX

MOTIVATION

- EARLY DETECTION OF FIRE
- DETECTION OF UNAUTHORIZED ACCESS TO ALARM HARDWARE
- WIRELESS INTEGRATION FOR [DOOR - WINDOW - MOTION] ALERTS USING ESP32

KEY FEATURES

- MULTI-SENSOR INTEGRATION
- STATE-DRIVEN OPERATION
- USER INTERFACE VIA KEYPAD AND LCD

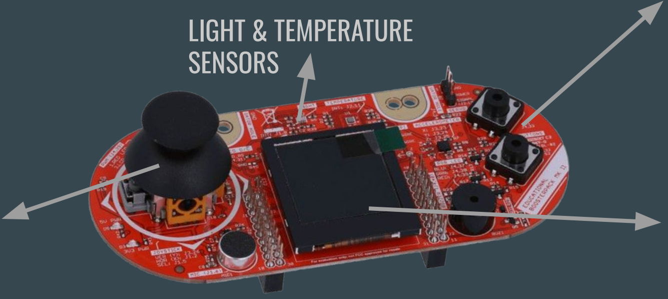


```
State_t evaluate_disarmed(){
    if (environment) {
        return TRIGGERED;
    } else if (password_correct && go_in_maintenance) {
        return MAINTENANCE;
    } else if (password_correct && go_in_armed) {
        return DELAY;
    } else if (password_correct && go_in_change_password) {
        return CHANGE_PASSWORD;
    }
    return DISARMED;
}
```

HARDWARE

MSP432

BOOSTERPACK

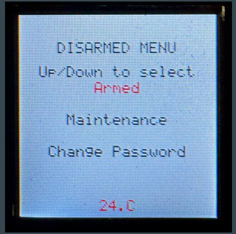


LIGHT & TEMPERATURE SENSORS

BUTTONS

JOYSTICK

LCD DISPLAY



DISARMED MENU

Up/Down to select

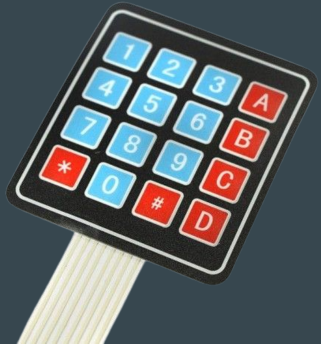
Armed

Maintenance

Change Password

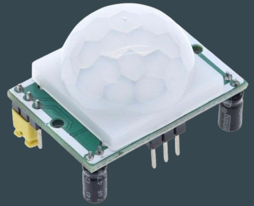
24.C

KEYPAD 4X4



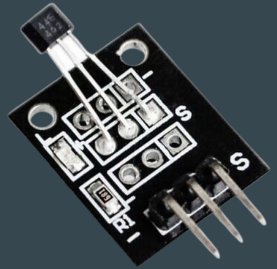
PIR SENSOR

MOVEMENT DETECTION



2 x MAGNETIC SENSORS

DOOR/WINDOW OPENING DETECTION



3 x ESP32 MODULES

MQTT

TWO DEPLOYMENT SOLUTIONS:

1. BROKER ON ESP32

✅ SELF-CONTAINED, NO EXTERNAL INFRASTRUCTURE

❌ LIMITED SCALABILITY, HIGHER POWER CONSUMPTION, SECURITY RISKS

⇒ BEST FOR SMALL-SCALE APPLICATIONS

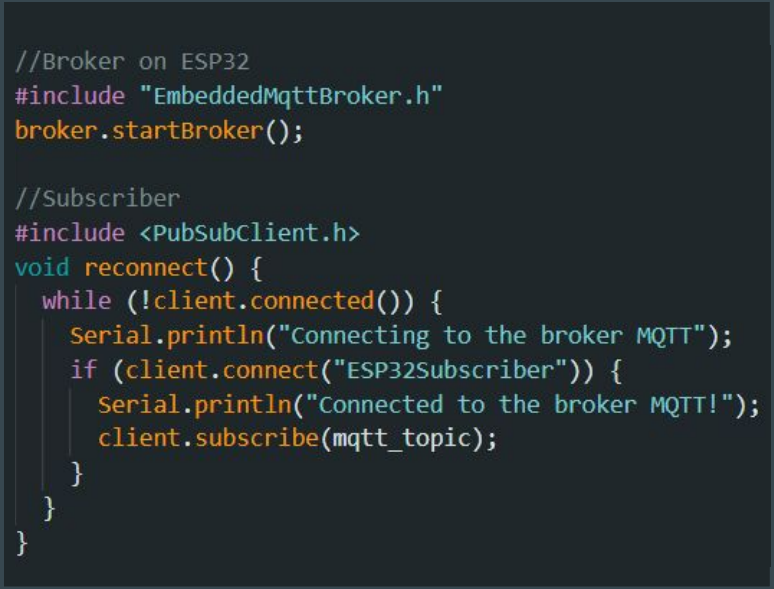
2. BROKER ON SEPARATE HOST (EMQX)

✅ HIGH PERFORMANCE, SECURE WITH TLS, SCALABLE

✅ SUPPORTS MILLIONS OF CONNECTIONS & ADVANCED MANAGEMENT FEATURES

❌ REQUIRES A SEPARATE HOST

⇒ IDEAL FOR LARGER IOT DEPLOYMENTS



```
//Broker on ESP32
#include "EmbeddedMqttBroker.h"
broker.startBroker();

//Subscriber
#include <PubSubClient.h>
void reconnect() {
  while (!client.connected()) {
    Serial.println("Connecting to the broker MQTT");
    if (client.connect("ESP32Subscriber")) {
      Serial.println("Connected to the broker MQTT!");
      client.subscribe(mqtt_topic);
    }
  }
}
```

webhooks

Node-RED

STATE MACHINE

STATES

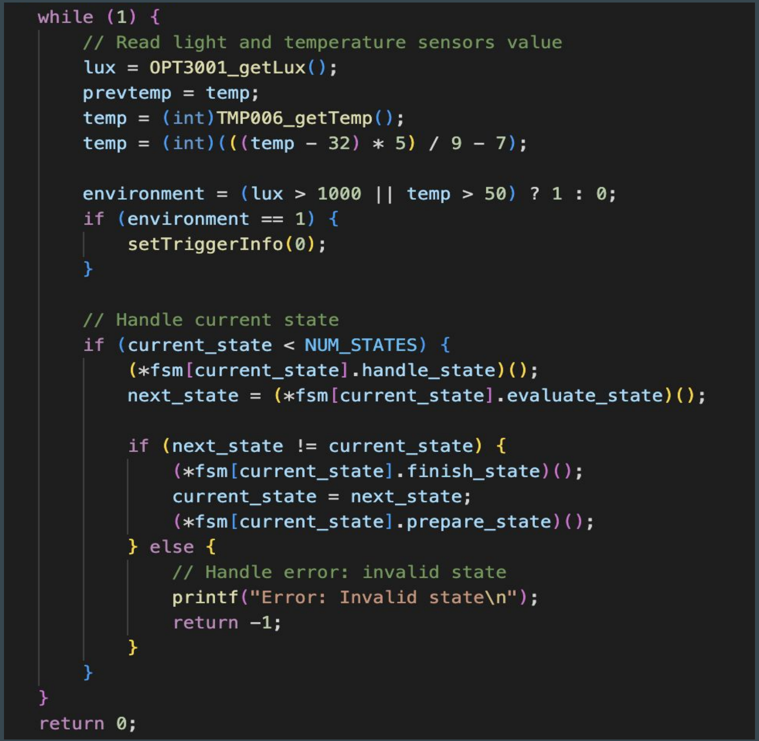
- DISARMED → System inactive - [GREEN]
- CHANGE PASSWORD → Change password
- MAINTENANCE → Sensors deactivated for maintenance work
- DELAY → Entry/exit period before armed - [GREEN]
- ARMED → Active monitoring - [RED]
- GRACE → Period after sensor trigger - [GREEN]
- TRIGGERED → Alarm active + buzzer - [BLUE]

HARDWARE CONTROL

- LED: Colours based on the current state
- Buzzer: Timer A0 PWM generation
- Environmental sensors integration

TRANSITION LOGIC

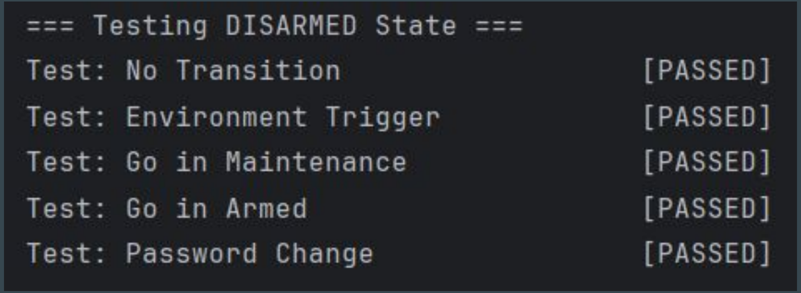
- State-specific handlers
- Sensor-triggered transitions through interrupts
- Password-protected modes



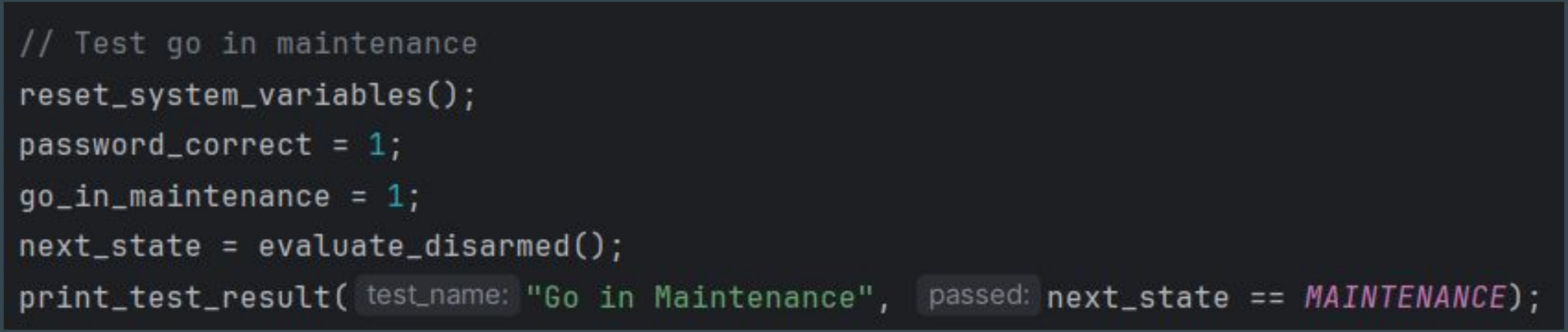
```c
while (1) {
    // Read light and temperature sensors value
    lux = OPT3001_getLux();
    prevtemp = temp;
    temp = (int)TMP006_getTemp();
    temp = (int)(((temp - 32) * 5) / 9 - 7);

    environment = (lux > 1000 || temp > 50) ? 1 : 0;
    if (environment == 1) {
        setTriggerInfo(0);
    }

    // Handle current state
    if (current_state < NUM_STATES) {
        (*fsm[current_state].handle_state)();
        next_state = (*fsm[current_state].evaluate_state)();

        if (next_state != current_state) {
            (*fsm[current_state].finish_state)();
            current_state = next_state;
            (*fsm[current_state].prepare_state)();
        } else {
            // Handle error: invalid state
            printf("Error: Invalid state\n");
            return -1;
        }
    }
}
return 0;
```

TESTING

- STANDALONE HOST-EXECUTABLE PROJECT WITH NO HARDWARE DEPENDENCIES
- COMPREHENSIVE FINITE STATE MACHINE LOGIC VALIDATION
- CONDITIONAL COMPILATION ISOLATES HARDWARE-INDEPENDENT CODE DURING TESTING



```
=== Testing DISARMED State ===
Test: No Transition                        [PASSED]
Test: Environment Trigger                  [PASSED]
Test: Go in Maintenance                    [PASSED]
Test: Go in Armed                          [PASSED]
Test: Password Change                      [PASSED]
```



```
// Test go in maintenance
reset_system_variables();
password_correct = 1;
go_in_maintenance = 1;
next_state = evaluate_disarmed();
print_test_result( test_name: "Go in Maintenance",  passed: next_state == MAINTENANCE);
```

FUTURE IMPROVEMENTS

ADDITIONAL AUTHENTICATION OPTIONS: FINGERPRINT SENSOR AND RFID

DISTINCTION BETWEEN CRITICAL SENSORS (WINDOW) AND NON-CRITICAL SENSORS (MAIN DOOR)

AUTHENTICATION SYSTEM TO REGISTER NEW SENSORS FOR IMPROVED SECURITY